Team:
Tarun Singh
Vennela Gandluri
Rohith Sali

GitHub names:
Tarun Singh (TarunSinghD)
Vennela Gandluri (vega5334)
Rohith Sali (Rohithsali)

Project Title: Leave Management System

1.Description:
A company has a large number of employees which includes full/part time employees and contractors. To smoothly manage the leave tracking operations of the company employees and to make the process error free and faster, the management has come up with a proposal to create an online, web based application called LMS (Leave Management System). Being an online application, LMS can be accessed through internet round the clock and is also an environment friendly way to avoid paper based processing of employee leave data and leave approval tracking details. The new system will greatly reduce the amount of time spent in leave processing because company's present system of manual tracking is very slow.

2.Features implemented:

User Requirements

ID	Requirement	Topic area	Actor
UR-001	As a employee, I should be able to select leave type	User account	Employee
UR-003	As a employee, I should be able to apply leave	User account	Employee
UR-004	As a employee, I should be able to login	User account	Employee
UR-005	As a employee, I should be able to view leave application status	User account	Employee
UR-006	As a employee, I should be able to cancel leave	Documentation	Employee

UR-007	As a employee, I should be able to update leave request	Documentation	Employee
UR-008	As a manager, I should be able to login	User account	Manager
UR-009	As a manager, I should be able to view leave request	User account	Manager
UR-010	As a manager, I should be able to approve leave	Documentation	Manager
UR-011	As a manager, I should be able to reject leave	Documentation	Manager
UR-012	As a HR, I should be able to login	User account	HR
UR-013	As a HR, I should be able to view leave approval	User account	HR
UR-015	As a HR, I should be able to add employee	Documentation	HR
UR-016	As a HR, I should be able to Delete employee		
UR-017	As a HR, I should be able to view all employees	User account	HR

Business Requirements

ID	**Requirement**	**Topic area**	**Actor**
BR-001	Username must be alphabets for registration	Authentication	HR, Employee, Manager
BR-003	All actors must have unique ID and name	User account	HR, Employee, Manager

BR-004	All employees must be able to apply leave	User account	HR, Employee, Manager
BR-005	All employees can select one type of leave	User account	HR, Employee, Manager

Functional requirements

ID	Requirement	Topic area	Actor
FR-002	Access should be granted to actors who can login	Authentication	HR, Employee, Manager
FR-004	As a system, all employees should be given the option to select type of leave	User account	HR, Employee, Manager

Non-Functional Requirements

ID	Requirement	Priority
NFR-001	Reliability - All data must be stored in a persistent and reliable	High
NFR-002	Security - account information must be stored in a secure manner	High
NFR-003	Performance - pages should load quickly	medium
NFR-004	Platform constraints - Functionality of the system should be same on all the platforms (Windows/mac)	low

3.Features not implemented:

User Requirements

ID	Requirement	Topic area	Actor
UR-014	As a HR, I should be able to update leave balance	Documentation	HR
UR-002	As a employee, I should be able to check leave balance	User account	Employee

Business Requirements

ID	Requirement	Topic area	Actor
BR-002	Password must contain atleast 6 characters	Authentication	HR, Employee, Manager

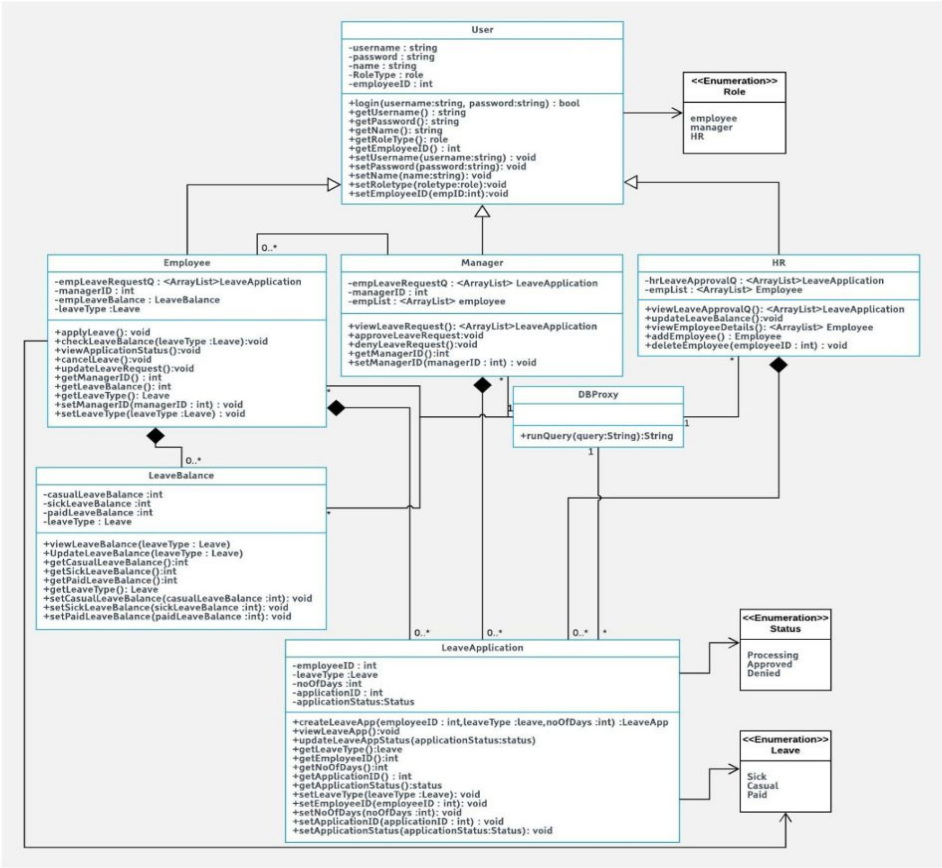
Functional Requirements

ID	Requirement	Topic area	Actor
FR-001	The system should mark initial status of leave applications as pending	Scheduling	System
FR-003	System should mark final status of leave application as completed	Scheduling	System

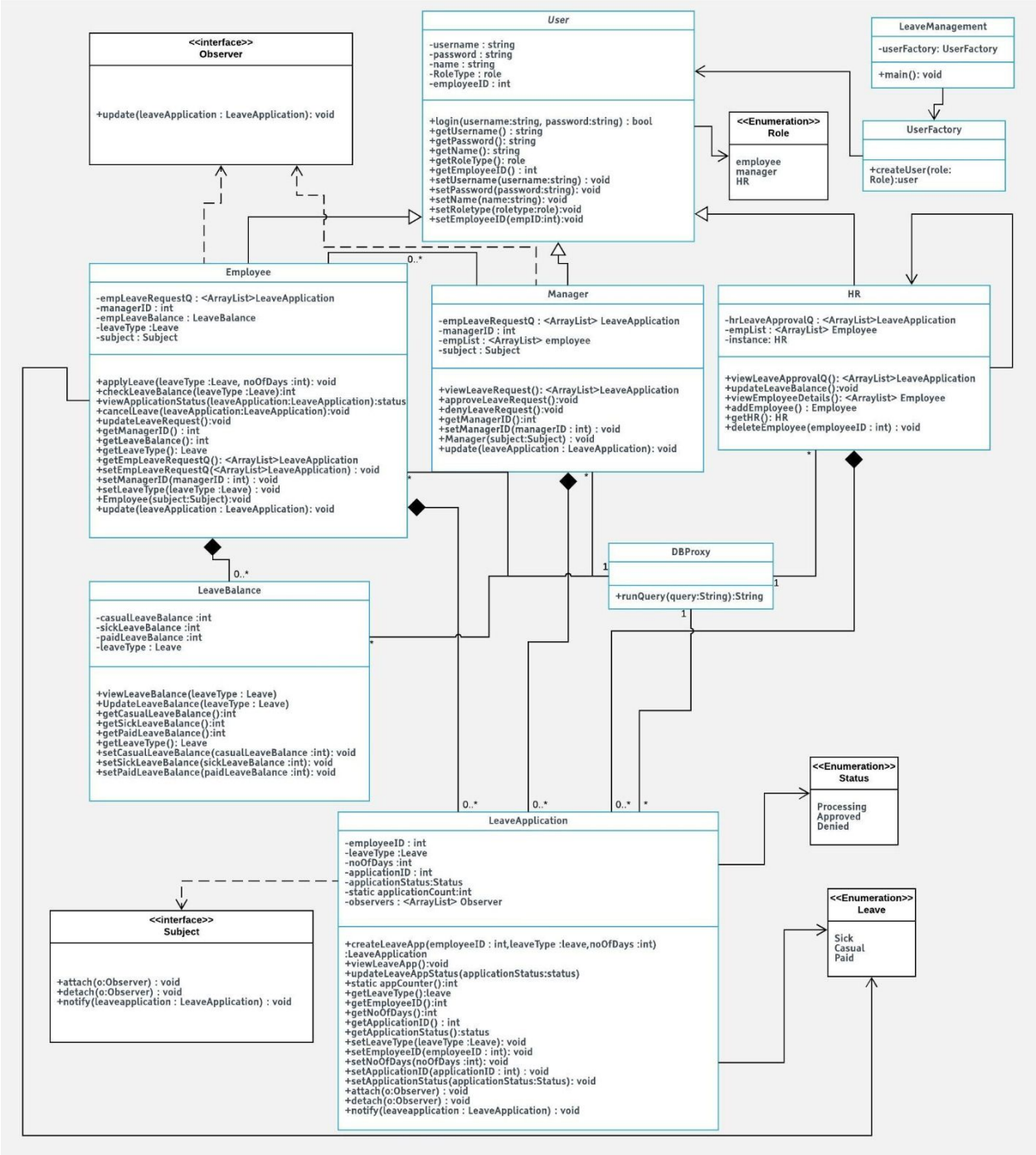
4.Previous class diagram (From part 2):

Link:https://github.com/TarunSinghD/Object-Oriented-Design-Final-Project/blob/master/Class_Diagram_Part2.jpeg

Class Diagram



Final class diagram:



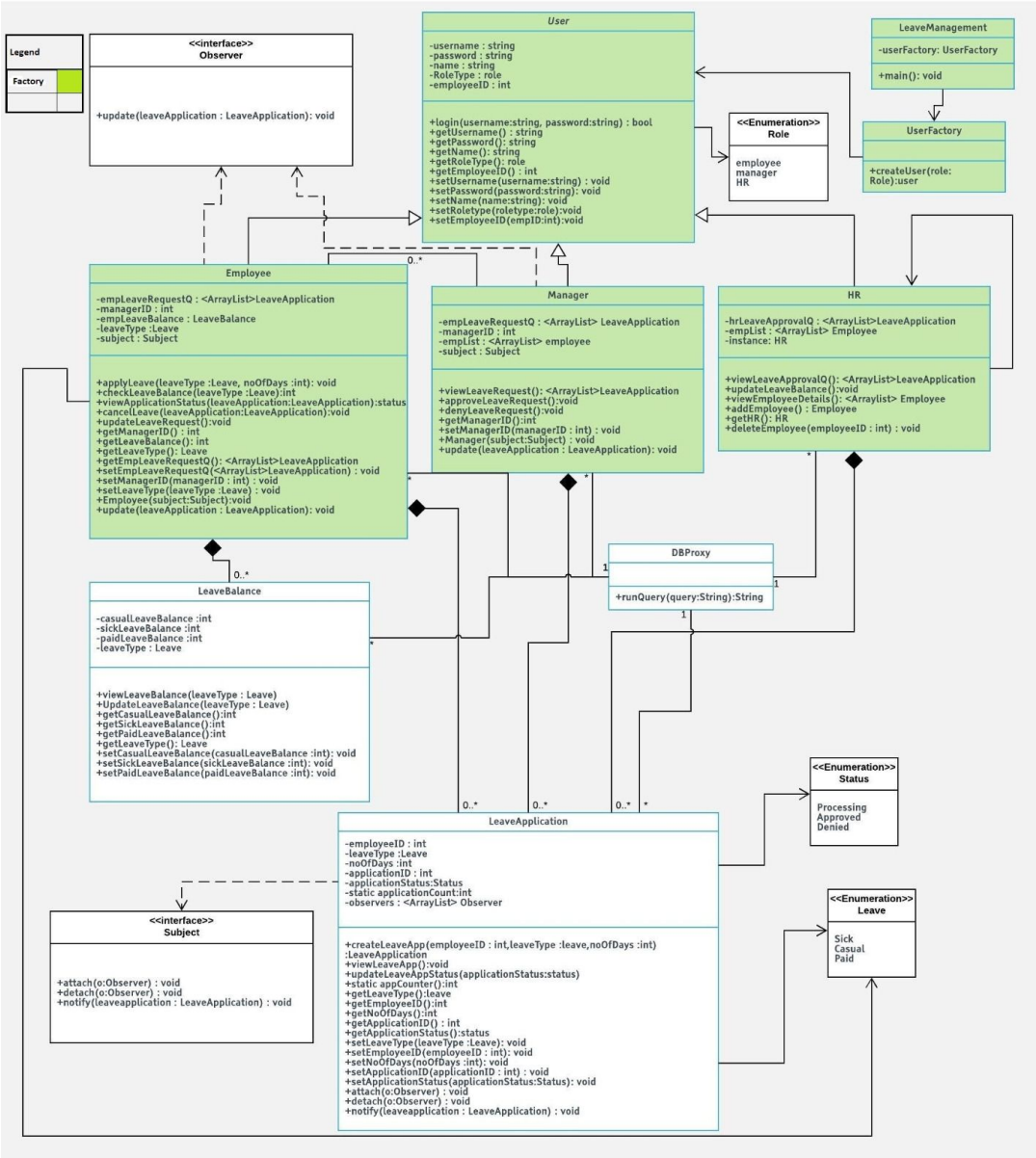
The main changes that were implemented from part 2 class diagram is :

- Implementation of the Observer, Factory and singleton design patterns.

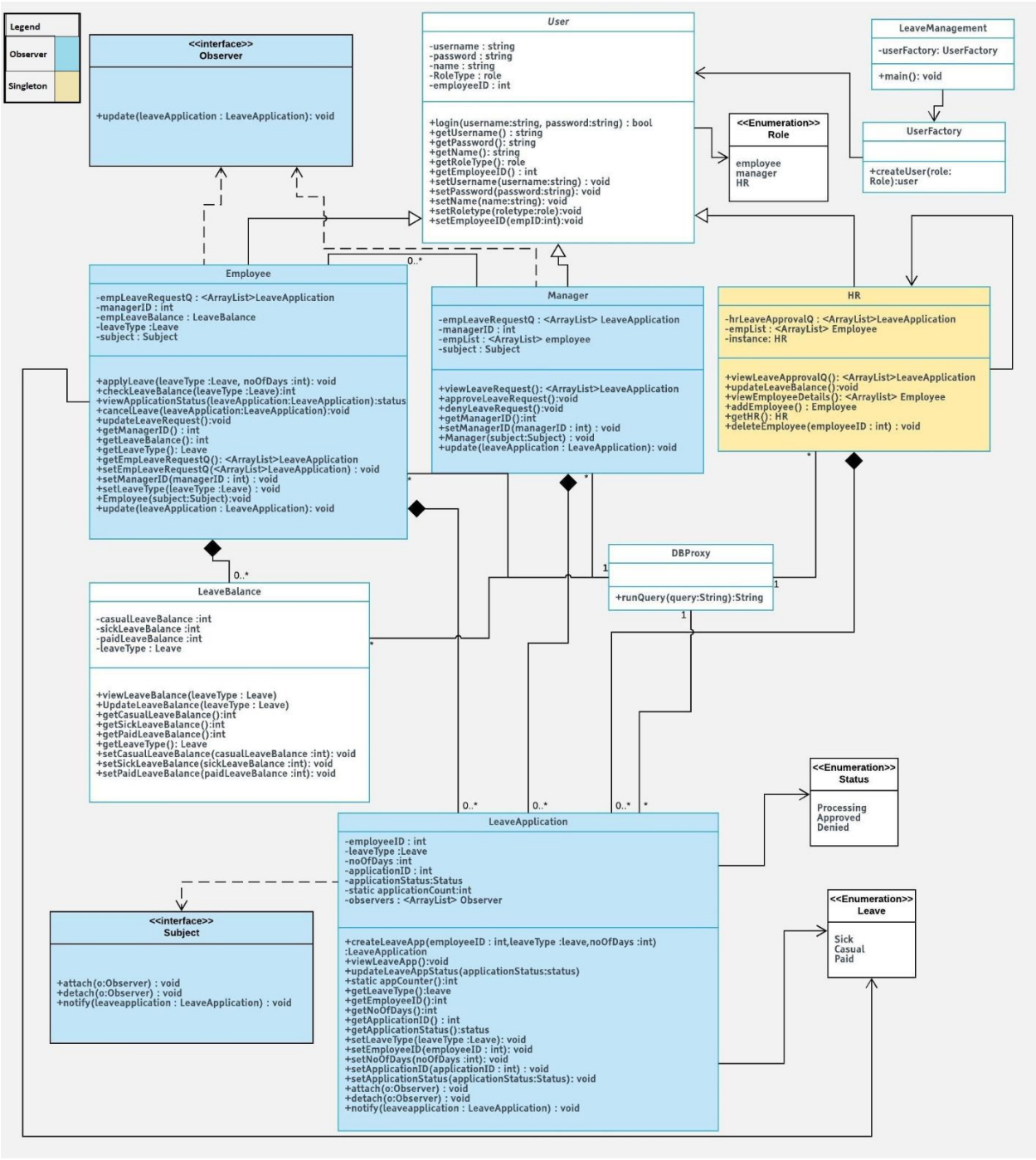
- Factory design pattern is used to implement polymorphism on User object - which can be instantiated to either Employee, Manager or HR by a factory class based on the role type.
- The Leave Application object has a “status” field associated with it. Employee should be notified when the status is changed to either approved or rejected. Manager should be notified when a “new” application is created.The observer design pattern fits well here with the publish-subscribe model.
- Singleton design pattern was implemented so that there is only one HR for the entire system.

5. Design pattern implementation in class diagram

1) Factory pattern highlighted



2) Observer and Singleton patterns highlighted



6.During the process of analysis and design,the lessons learnt were:

1. Use case diagram helped us learn the interaction of actors with the system
2. Class diagram was very pivotal in giving a good picture of what has to comprise our classes, and how different classes are connected to each other and database.
3. Activity diagrams gave a good understanding of the system flow.
4. Good understanding of design patterns and how they can be applied to our existing class diagram.
5. Implementation of a complete web application with MVC, with the help of Java servlets and JSP.